Running Multiple Serial Jobs to Reduce Walltime

Category: Effective Use of PBS

DRAFT

This article is being reviewed for completeness and technical accuracy.

On Pleiades, running multiple serial jobs within a single batch job can be accomplished with following example PBS scripts. The maximum number of processes you can run on a single node will be limited to the core-count-per-node or the maximum number that will fit in a given node's memory, whichever is smaller.

```
processor type  cores/node      available memory/node
 Harpertown         8                     7.6 GB
 Nehalem-EP         8                    22.5 GB
 Westmere-EP       12                    22.5 GB
```

The examples below allow you to spawn serial jobs accross nodes using the mpiexec command. Note that a special version of mpiexec from the mpi-mvapich2/1.4.1/intel module is needed in order for this to work. This mpiexec keeps track of $PBS_NODEFILE and places each serial job onto the CPUs listed in $PBS_NODEFILE properly. The use of the arguments "-comm none" for this version of mpiexec is essential for serial codes or scripts. In addition, to launch multiple copies of the serial job at once, the use of the mpiexec-supplied $MPIEXEC_RANK environment variable is needed to distinguish different input/output files for each serial job. This is demonstrated with the use of a wrapper script "wrapper.csh" in which the input/output identifier (i.e., ${rank}) is calculated from the sum of $MPIEXEC_RANK and an argument provided as input by the user.

Example 1:

This first example runs 64 copies of a serial job, assuming that 4 copies will fit in the available memory on one node and 16 nodes are used.

serial1.pbs:

```
#PBS -S /bin/csh
#PBS -j oe
#PBS -l select=16:ncpus=4
#PBS -l walltime=4:00:00

module load mpi-mvapich2/1.4.1/intel

cd $PBS_O_WORKDIR
```

```
mpiexec -comm none -np 64 wrapper.csh 0
```

wrapper.csh:

```
#!/bin/csh -f
@ rank = $1 + $MPIEXEC_RANK
./a.out < input_${rank}.dat > output_${rank}.out
```

This example assumes that input files are named input_0.dat, input_1.dat, ... and that they are all located in the directory where the PBS script is submitted from (i.e., $PBS_O_WORKDIR). If the input files are in different directories, then wrapper.csh can be modified appropriately to cd into different directories as long as the directory names are differentiated by a single number that can be obtained from $MPIEXEC_RANK (=0, 1, 2, 3, ...). In addition, be sure that wrapper.csh is executable by you and you have the current directory included in your path.

Example 2:

A second example provides the flexibility where the total number of serial jobs may not be the same as the total number of CPUs requested in a PBS job. Thus, the serial jobs are divided into a few batches and the batches are processed sequentially. Again, the wrapper script is used where multiple versions of the program "a.out" in a batch are run in parallel.

serial2.pbs:

```
#PBS -S /bin/csh
#PBS -j oe
#PBS -l select=10:ncpus=3
#PBS -l walltime=4:00:00

module load mpi-mvapich2/1.4.1/intel

cd $PBS_O_WORKDIR

# This will start up 30 serial jobs 3 per node at a time.
# There are 64 jobs to be run total, only 30 at a time.

# The number to run in total defaults here to 64 or the value
# of PROCESS_COUNT that is passed in via the qsub line like:
# qsub -v PROCESS_COUNT=48 serial2.pbs
#

# the total number to run at once is automatically determined
# at runtime by the number of cpus available.
# qsub -v PROCESS_COUNT=48 -l select=4:ncpus=3 serial2.pbs
# would make this 12 per pass not 30. no changes to script needed.

if ( $?PROCESS_COUNT ) then
 set total_runs=$PROCESS_COUNT
else
 set total_runs=64
endif
```

```
set batch_count=`wc -l < $PBS_NODEFILE`

set count=0

while ($count < $total_runs)
  @ rank_base = $count
  @ count += $batch_count
  @ remain = $total_runs - $count
  if ($remain < 0) then
    @ run_count = $total_runs % $batch_count
  else
    @ run_count = $batch_count
  endif
  mpiexec -comm none -np $run_count wrapper.csh $rank_base
end
```

Article ID: 184
Last updated: 03 Aug, 2011
Computing at NAS -> Running Jobs with PBS -> Effective Use of PBS -> Running Multiple Serial Jobs to Reduce Walltime
http://www.nas.nasa.gov/hecc/support/kb/entry/184/?ajax=1